Prof. Dr.-Ing. Ralf Steinmetz
Multimedia communications Lab

Dipl. Inf. Robert Konrad
Polona Caserman, M.Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Game Technology
Winter Semester 2017/2018

Solution 8

General Information

- The exercises may be solved by teams of up to three people.
- The solutions have to be uploaded to the Git repositories assigned to the individual teams.
- **The submission date (for practical and theoretical tasks) is noted on top of each exercise sheet.**
- If you have questions about the exercises write a mail to game-technology@kom.tu-darmstadt.de or use the forum at https://www.fachschaft.informatik.tu-darmstadt.de/forum/viewforum.php?f=557

P8. Practical Tasks: Physics (5 Points)

In this exercise, the overall task is to build a demo in which you can shoot spheres from the camera position by pressing the space key and have them interact with each other and a plane. The code is provided for you; your task is to fill out the respective functions. The code can be found at https://github.com/TUDGameTechnology/Exercise8.git

Please remember to push into a branch called "exercise8".

You can find the practical solutions at https://github.com/TUDGameTechnology/Solution8.git.

P8.1 Particle Systems – Billboards (1 point)

You can see in the exercise code that the particles are being spawned, but when the camera turns, they are visible from the side and back. Instead, we want them oriented towards the camera. Implement this rotation. The view matrix of the camera is available to the particle system.

P8.2 Particle Systems – Control parameters (2 points)

Decide on an effect you want to create using particle systems that uses one control parameter that is not present in the code you are given. For example, you can implement the "fire" example from the slides. Other control parameter could be the rotation of the particle billboards (you might want to change the texture in this case as the provided texture is symmetrical), the size of the billboards or the mass. For rain, you could spawn a "splash" particle when the rain hits the ground.

P8.3 Numerical Integration (1 point)

Implement an Euler integrator for the physics computations. The necessary function can be found in "PhysicsObject.cpp". Note that it is beneficial to multiply the resulting velocity during the integration step with a damping coefficient. This coefficient accounts for energy that is normally lost when objects move and interact, and helps the system come to rest eventually.

P8.4 Sphere-Sphere Intersections (1 point)

Implement an intersection test for two spheres with each other. This task entails implementing the functions
bool IntersectsWith(const SphereCollider& other);
vec3 GetCollisionNormal(const SphereCollider& other);
float PenetrationDepth(const SphereCollider& other);

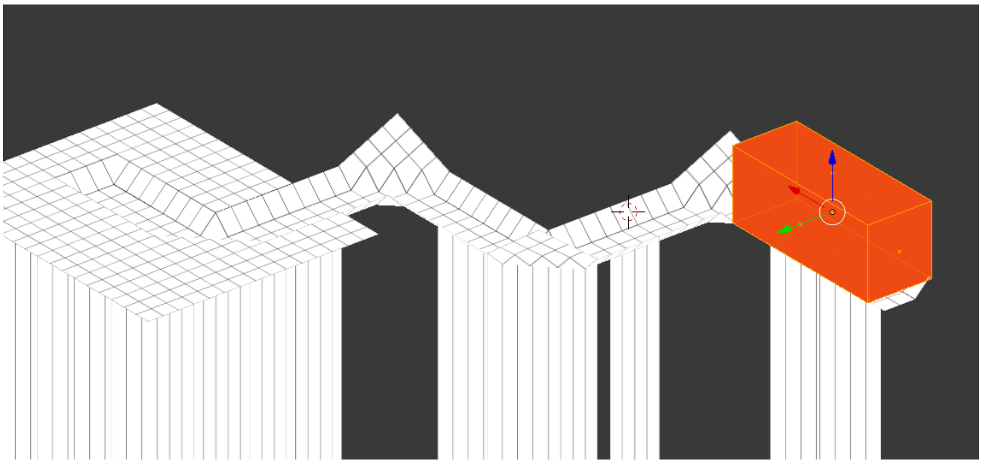
which are found in "Collision.h"

T8. Theoretical Tasks: Physics (5 Points)

T8.1 Sphere-Box-Intersection (3 point)

Research a method for intersection between a box and a sphere or derive your own.

Describe the chosen intersection test and write it down in pseudocode.



See for example: http://theorangeduck.com/page/correct-box-sphere-intersection

The test is composed of two sub-tests:

1) Checking if the sphere is inside the box

This can be done by checking all the planes of the faces of the box, and checking that the sphere is on the inside of all of them.

This can be checked with the sphere-plane distance check. The formula for the signed distance of the sphere to the plane was:

$$distance = x \cdot n - d$$

where x is the center of the sphere, n is the plane normal and d is the distance from the origin, along this normal.

Therefore, we need to check distance < radius for each plane.

2) Checking whether the sphere intersects a side of the box

It is not sufficient to test for intersection with the planes, since the planes extend beyond the box faces they are defined by, so an intersection with a plane might be outside of the face.

This can be checked with the results of the plane tests. We can check whether the intersection is within the actual box side and not just with the plane.

Figures 1 to 4 below show how a false positive can come up if we only test each plane for intersection and don't check if the intersection is within the box side.

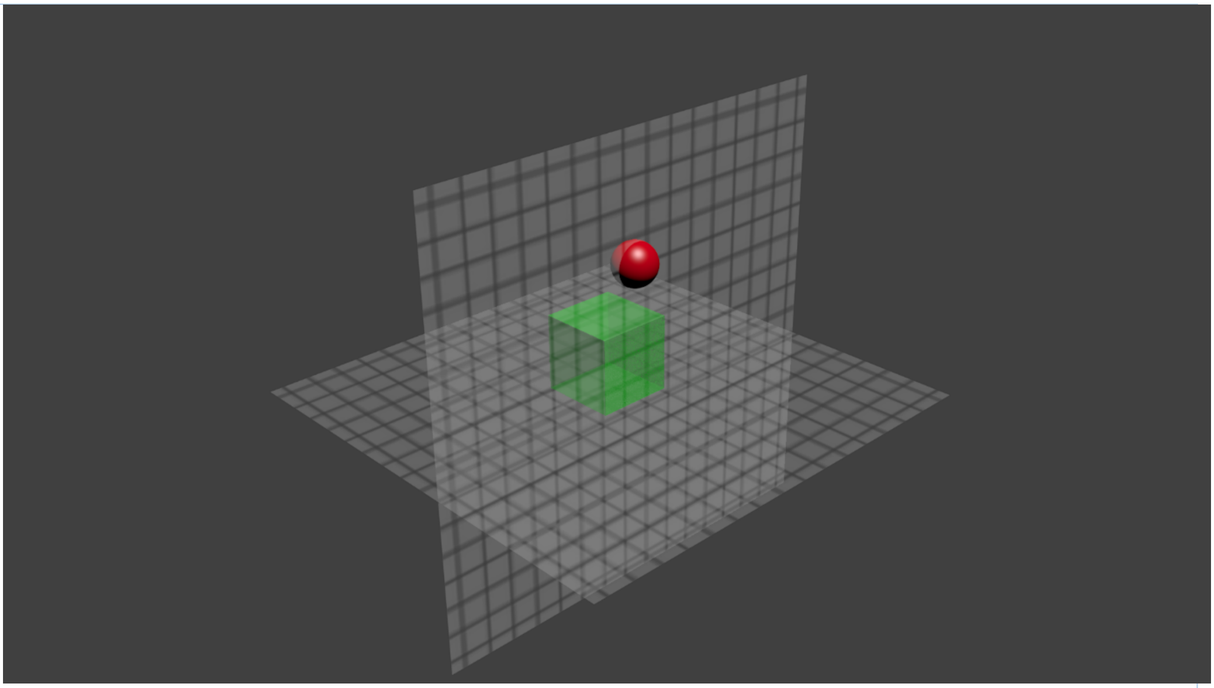


Figure 1: A false positive where the sphere is intersecting one side's plane, but not the actual face.

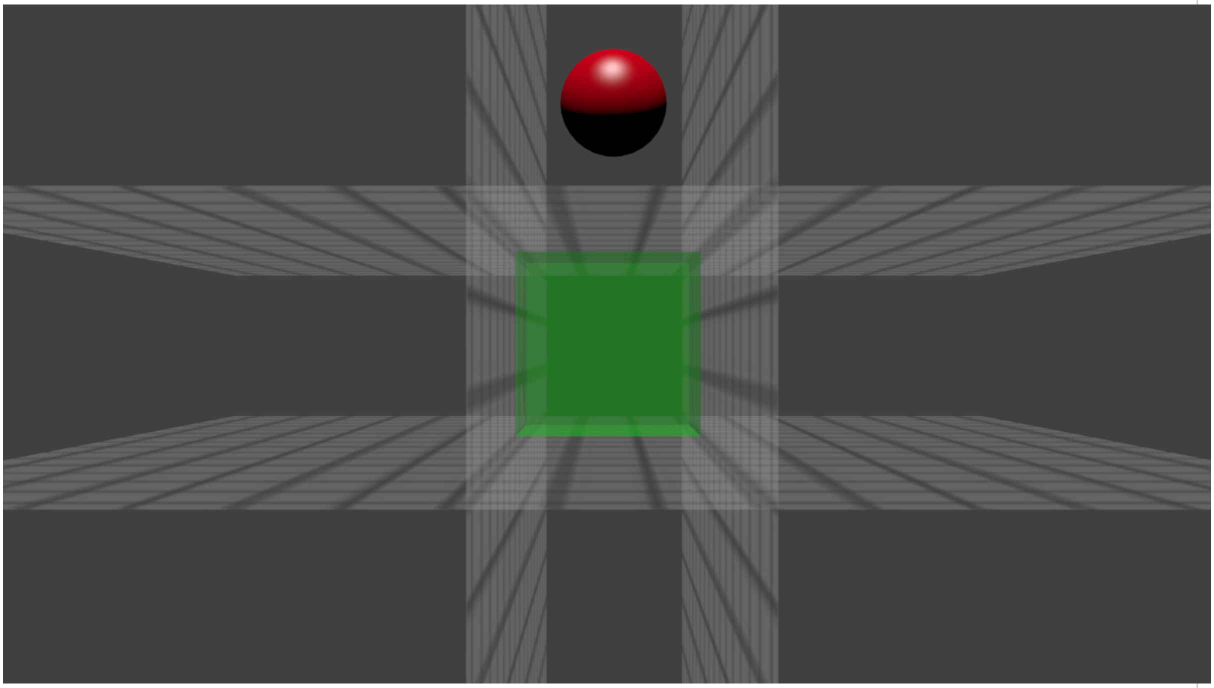


Figure 2: The situation from figure 1 from the side. We see that the sphere is actually outside the planes that delimit the box face in question.

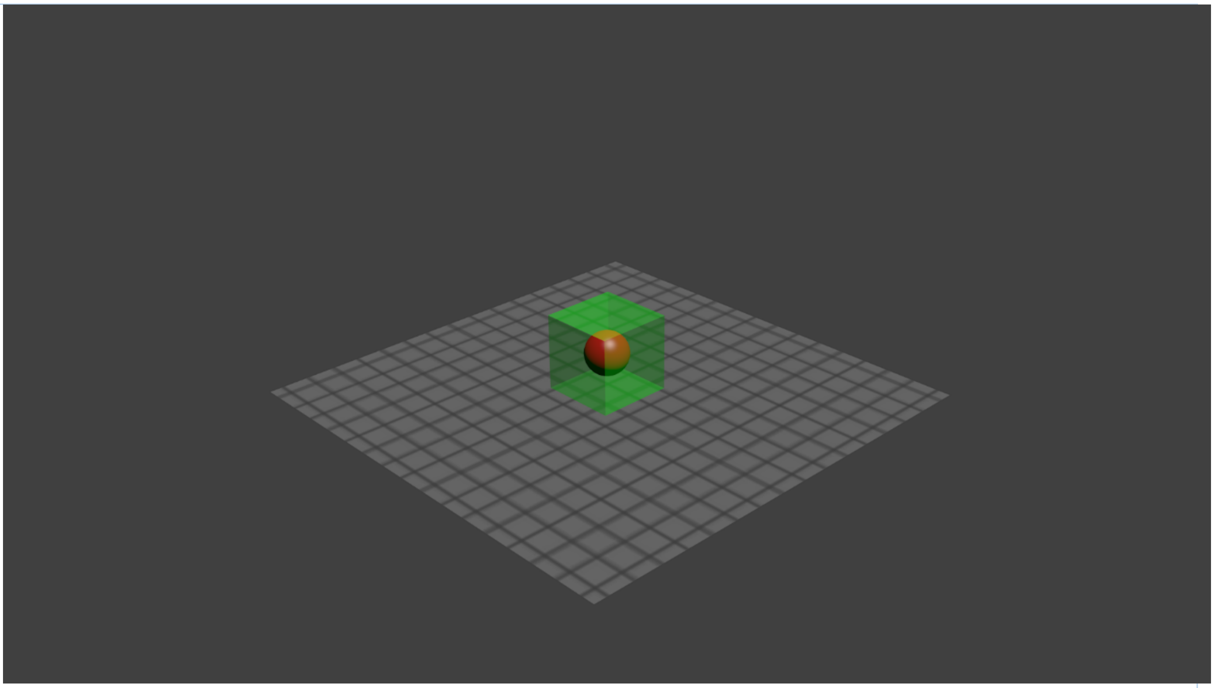


Figure 3: The sphere is inside the box.

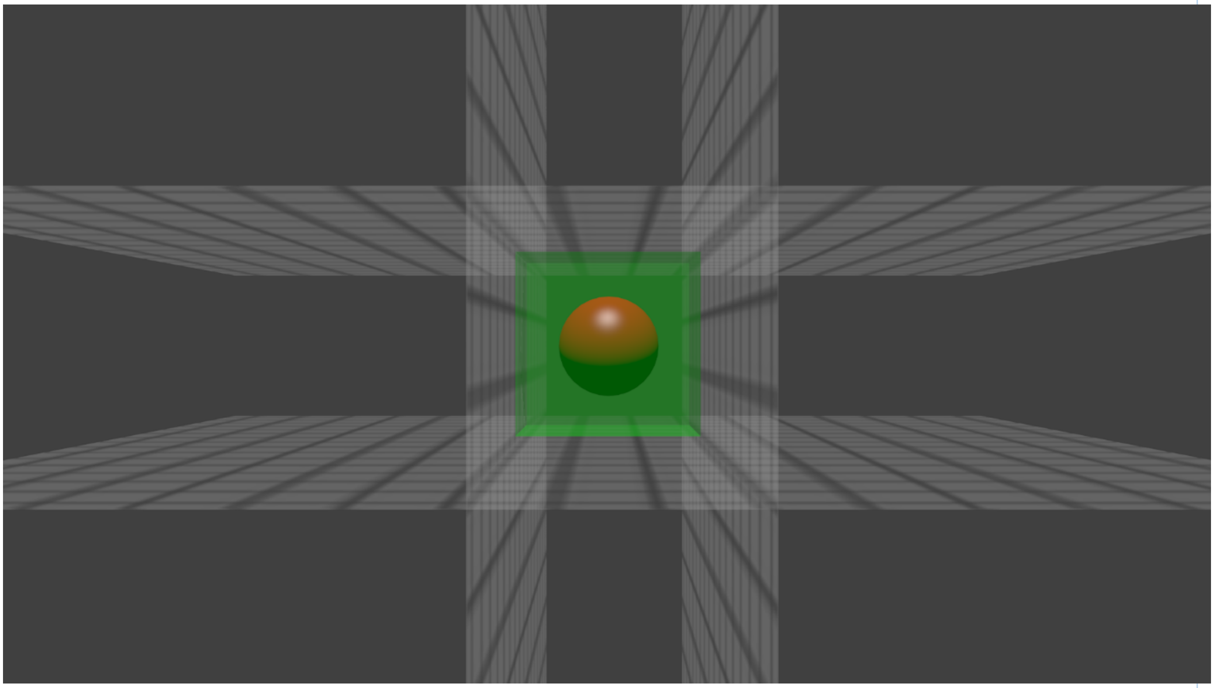


Figure 4: The situation from figure 3 from the side. We see that the sphere is inside of all planes delimiting the box side.

T8.2 Integration constant acceleration (2 Points)

Imagine a sphere with mass m=1kg that is falling down in a vacuum. At t=0, the sphere is released. The velocity v at t=0 is 0. The acceleration a is constant at 10 m/s² in the negative y-direction.

Calculate the movement of the sphere by integrating the equations of motion using the Euler method as explained in the lecture (you can use a spreadsheet application or a script).

a) Use a time step of deltaT=1s and provide the values of height (z) and velocity (v) for the situation at time t=3s.
b) Use a time step of deltaT = 0.5s and provide the values. Compare the results with an analytical solution and discuss them.

a)

t	y	v	acceleration	mass	deltaT
0	0	0	-10	1	1
1	-10	-10	-10	1	1
2	-30	-20	-10	1	1
3	**-60**	**-30**	**-10**	**1**	**1**
4	-100	-40	-10	1	1
5	-150	-50	-10	1	1

b)

t	y	v	acceleration	mass	deltaT
0	0	0	-10	1	0,5
0,5	-2,5	-5	-10	1	0,5
1	-7,5	-10	-10	1	0,5
1,5	-15	-15	-10	1	0,5
2	-25	-20	-10	1	0,5
2,5	-37,5	-25	-10	1	0,5
3	**-52,5**	**-30**	**-10**	**1**	**0,5**
3,5	-70	-35	-10	1	0,5
4	-90	-40	-10	1	0,5
4,5	-112,5	-45	-10	1	0,5
5	-137,5	-50	-10	1	0,5

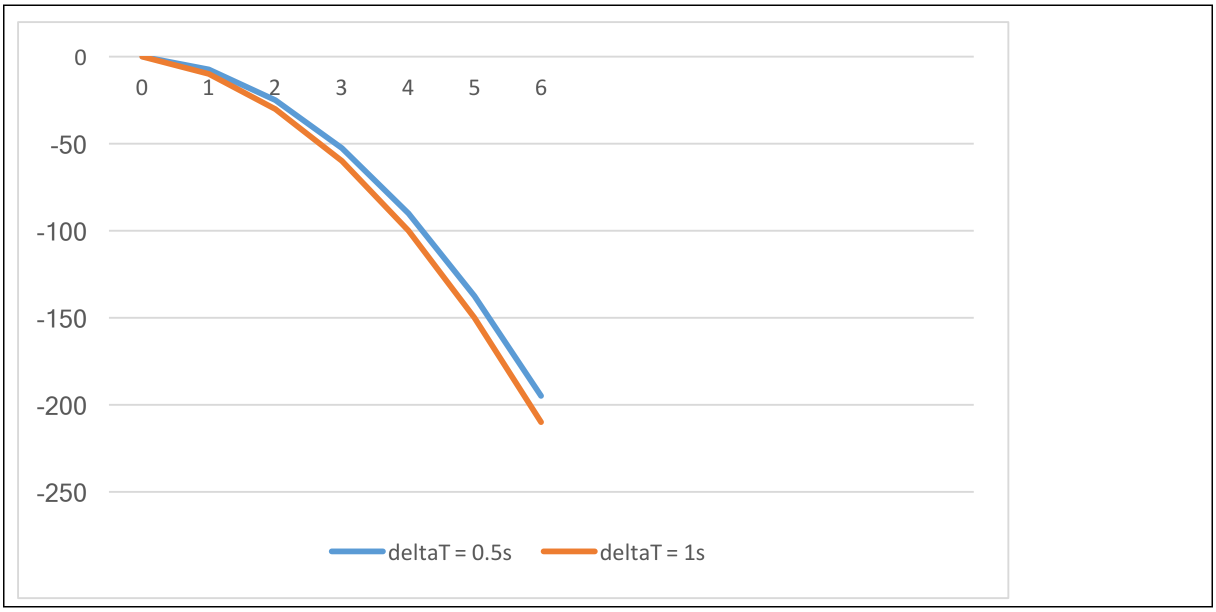
We can find an analytical solution by the formula

$$y(t) = v_0 t + \frac{1}{2} a t^2$$

Therefore,

$$y(3) = 0 \cdot 3 + \frac{1}{2} 10 \cdot 3^2$$
$$= \frac{1}{2} 10 \cdot 9 = 45$$

We see that the shorter time step could better approximate the correct value.


0
-50
-100
-150
-200
-250
0
1
2
3
4
5
6
deltaT = 0.5s
deltaT = 1s